The Solution of Singular Value Problems Using Systolic Arrays

Richard P. Brent†
Franklin T. Luk*

TR 84-626
August 1984

†Centre for Mathematical Analysis
Australian National University
Canberra, A.C.T. 2601
Australia

*Department of Computer Science
Cornell University
Ithaca, New York 14853

---

*The work of this author was supported in part by the National Science Foundation under grant MCS-8213718.

The Solution of Singular Value Problems Using Systolic Arrays

Richard P. Brent

Centre for Mathematical Analysis
Australian National University
Canberra, A.C.T. 2601
Australia

Franklin T. Luk

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.

ABSTRACT

This paper concerns the computation of the singular value decomposition using systolic arrays. Two different linear time algorithms are presented.

Keywords and Phrases: Systolic arrays, QR-decomposition, singular value decomposition, Jacobi algorithms, real-time computation, VLSI.

Invited paper to appear in *Proceedings of SPIE Volume 495, Real Time Signal Processing VII* (1984).

Introduction

Perhaps the most important factorization of a given $m \times n$ matrix A $(m \geq n)$ is its singular value decomposition (SVD):

$$A = U \Sigma V^T , \tag{1}$$

where the matrices U $(m \times m)$ and V $(n \times n)$ are orthogonal, and the matrix Σ $(m \times n)$ is nonnegative diagonal. For details on applications of the SVD see Golub and Luk[1] and Golub and Van Loan[2]. The best sequential SVD algorithm (due to Golub) is coded in LINPACK[3]. Recently, there has been much interest in computing the SVD using systolic arrays, principally due to the needs of real time signal processing (Bromley and Speiser[4]). SVD arrays are presented in Brent and Luk[5], Brent, Luk and Van Loan[6], Finn, Luk and Pottle[7], Heller and Ipsen[8], Luk[9], and Schreiber[10].

The fastest SVD algorithms (effectively linear time) are the Jacobi procedures of Brent et al.[6] and Luk[9]. Jacobi-type methods are natural for matrix computations using processor arrays: they have been proposed for the symmetric eigenvalue decomposition by Brent and Luk[5], for the QR-decomposition by Luk[11], and for the Schur decomposition by Stewart[12]. In addition, the methods used for finding eigenvalues and singular values on the first parallel computer, the ILLIAC IV, were also of the Jacobi type (Luk[13] and Sameh[14]). Unfortunately, Jacobi-SVD algorithms are applicable only to square matrices. For an $m \times n$ matrix A, an obvious strategy is to first compute its QR-decomposition (QRD):

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} , \tag{2}$$

where the matrix Q $(m \times m)$ is orthogonal and the matrix R $(n \times n)$ is upper triangular, and then apply an SVD procedure to R. This approach is particularly suitable for the case where $m >> n$ (cf. Chan[15]). QRD-arrays have been thoroughly studied; see Ahmed, Delosme and Morf[16], Bojanczyk, Brent and Kung[17], Gentleman and Kung[18], Heller and Ipsen[19], Johnsson[20], Luk[11] and Sameh[21]. However, the interfacing of QRD and SVD arrays can be a difficult problem. In fact, the QRD algorithm in Luk[11] is the only algorithm implementable on the square SVD array of Brent et al.[6]. Recently, Luk[9] presents the only triangular processor array that can compute both the QRD and the SVD.

The purpose of this paper is to survey the two linear-time SVD methods[6,9] and their associated processor arrays.

Eigenvalue decomposition

The classical method of Jacobi uses a sequence of plane rotations to

diagonalize a symmetric matrix A. Let us denote a Jacobi rotation of an angle θ in the (i,j) plane by $J(i,j,\theta) \equiv J$, where $i<j$. The matrix J is the same as the identity matrix except for four strategic elements:

$$J_{ii} = c \ , \ J_{ij} = s \ , \tag{3}$$
$$J_{ji} = -s \ , \ J_{jj} = c \ ,$$

where $c = \cos\theta$ and $s = \sin\theta$. Setting $B = J^T A J$, we get

$$\begin{bmatrix} b_{ii} & b_{ij} \\ b_{ji} & b_{jj} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} . \tag{4}$$

If we choose the cosine-sine pair (c,s) such that

$$b_{ij} = b_{ji} = a_{ij}(c^2-s^2) + (a_{ii}-a_{jj})cs = 0 \ , \tag{5}$$

then B becomes "more diagonal" than A in the sense that

$$off(B) = off(A) - 2a_{ij}^2 \ , \tag{6}$$

where

$$off(C) \equiv \sum_{p \neq q} c_{pq}^2 \quad \text{for } C = (c_{pq}) \ . \tag{7}$$

Jacobi methods for the symmetric eigenproblem are of interest because they lend themselves to parallel computations. Brent and Luk[5] have developed a square processor that can diagonalize an $n \times n$ symmetric matrix in effectively $O(n)$ time. It may seem that software (or hardware) for the symmetric eigenvalue problem can be used to solve the SVD problem. For example, we may compute the eigenvalue decomposition

$$V^T(A^TA)V = \text{diag}(\sigma_1^2, \cdots, \sigma_n^2) \ , \tag{8}$$

where $V = (v_1, \cdots, v_n)$ is orthogonal and the σ_i satisfy

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0 \ , \tag{9}$$

with $r = \text{rank}(A)$. We next calculate the vectors

$$u_i = (1/\sigma_i)Av_i \quad (i=1, \cdots, r) \ , \tag{10}$$

and determine the others: $\{u_{r+1}, \cdots, u_m\}$ so that the matrix $U = (u_1, \cdots, u_m)$ is orthogonal. The factorization $U^TAV = \text{diag}(\sigma_1, \cdots, \sigma_n)$ gives an SVD of A. Thus, one can theoretically compute an SVD of A via an eigenvalue decomposition of A^TA. Unfortunately, well-known numerical difficulties are associated with the explicit formation of A^TA.

A way round this difficulty is to apply the Jacobi method implicitly. This is the gist of the "one-sided" Hestenes[22] approach in which the matrix V is determined so that the columns of AV are mutually orthogonal. Implementations are discussed in Luk[13] and in Brent and Luk[5]. In the latter reference a systolic

array is developed that is tailored to the method. However, inner products of m-vectors are required for each (c,s) computation. Because of this, the speed of their parallel algorithm is effectively $O(mn)$ for a linear array of $O(n)$ processors, and $O(n\log m)$ for a two-dimensional array of $O(mn)$ processors with some special interconnection patterns for inner-product computations. Another drawback of the one-sided Jacobi method is that it also does not directly generate the vectors $u_{r+1}, \cdots, u_m$. This is an inconvenience in the systolic array setting since one would need a special architecture to carry out the matrix-vector multiplications in (10).

Another approach to the SVD problem is to compute an eigenvalue decomposition of the $(m+n)\times(m+n)$ symmetric matrix

$$C = \begin{bmatrix} O & A \\ A^T & 0 \end{bmatrix}. \tag{11}$$

Note that if

$$\begin{bmatrix} O & A \\ A^T & 0 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = \sigma \begin{bmatrix} u \\ v \end{bmatrix}, \tag{12}$$

then $A^TAv = \sigma^2 v$ and $AA^Tu = \sigma^2 u$. Thus, the eigenvectors of C are "made up" of the singular vectors of A. It can also be shown that the spectrum of C is given by

$$\lambda(C) = \{ \pm\sigma_1, \cdots, \pm\sigma_n, 0, \cdots, 0 \}. \tag{13}$$

The disadvantages of this approach are that C has expanded dimension and that recovering the singular vectors may be a difficult numerical task. In addition, the case of rank$(A)<n$ requires extra work to generate $v_{r+1}, \cdots, v_n$.

To summarize, it is preferable from several different points of view *not* to approach the SVD problem as a symmetric eigenvalue problem.

Two-by-two SVD

The basic tool in a Jacobi-SVD method is the 2×2 plane rotation

$$J(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{14}$$

as the basic problem concerns the diagonalization of a 2×2 matrix by the rotations $J(\theta)$ and $K(\phi)$:

$$J(\theta)^T \begin{bmatrix} w & x \\ y & z \end{bmatrix} K(\phi) = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}. \tag{15}$$

A two-stage procedure is adopted. First, find a rotation $S(\psi)$ to symmetrize B:

$$S(\psi)^T \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} p & q \\ q & r \end{bmatrix} . \tag{16}$$

If $x=y$ we choose $\psi=0$, otherwise we compute

$$\begin{aligned} \rho &= \frac{w+z}{x-y} \equiv \mathrm{ctn}\psi , \\ \sin\psi &= \frac{\mathrm{sign}(\rho)}{\sqrt{1+\rho^2}} , \\ \cos\psi &= \rho \sin\psi . \end{aligned} \tag{17}$$

Second, diagonalize the result:

$$K(\phi)^T \begin{bmatrix} p & q \\ q & r \end{bmatrix} K(\phi) = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} . \tag{18}$$

Suppose $q \neq 0$ (else choose either $\phi=0$ or $\phi=\pi/2$). It is well known that $t \equiv \tan\phi$ satisfies the quadratic equation:

$$t^2 + 2\rho t - 1 = 0 , \tag{19}$$

where

$$\rho = \frac{r-p}{2q} \equiv \mathrm{ctn}2\phi . \tag{20}$$

The two solutions to (19) are

$$\begin{aligned} t &= \frac{\mathrm{sign}(\rho)}{|\rho| + \sqrt{1+\rho^2}} , \\ \cos\phi &= \frac{1}{\sqrt{1+t^2}} , \\ \sin\phi &= t \cos\phi \end{aligned} \tag{21}$$

and

$$\begin{aligned} t &= -\mathrm{sign}(\rho) \left[|\rho| + \sqrt{1+\rho^2} \right] , \\ \cos\phi &= \frac{1}{\sqrt{1+t^2}} , \\ \sin\phi &= t \cos\phi . \end{aligned} \tag{22}$$

The angle ϕ associated with (21) is the smaller of the two possibilities; it satisfies $0 \leq |\phi| < \pi/4$, whereas the one associated with (22) satisfies $\pi/4 \leq |\phi| < \pi/2$. We refer to a rotation through the smaller angle as an "inner rotation" and one through the larger angle as an "outer rotation". The "inner rotation" is chosen in Brent et al.[6] and the "outer rotation" in Luk[9]. If the given matrix is diagonal ($x=y=0$) then an "inner rotation" means $\phi=0$ and an

"outer rotation" implies $\phi=\pi/2$. In the former case the matrix stays unchanged, whereas in the latter case the singular values are interchanged:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} w & 0 \\ 0 & z \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} z & 0 \\ 0 & w \end{bmatrix} . \tag{23}$$

Finally, $J(\theta)$ is given by

$$J(\theta)^T = K(\phi)^T S(\psi)^T , \tag{24}$$

i.e., $\theta = \phi + \psi$.

By solving an appropriate sequence of 2×2 SVD problems, we compute an SVD of a general $n \times n$ matrix A. The Jacobi transformation is

$$T_{ij} \ : \ A \leftarrow J_{ij}^T A \ K_{ij} , \tag{25}$$

where J_{ij} and K_{ij} are rotations in the (i,j) plane chosen to annihilate the (i,j) and (j,i) elements of A. As in the symmetric case, the transformation T_{ij} will produce a matrix B satisfying

$$off(B) = off(A) - r_{ij}^2 - r_{ji}^2 , \tag{26}$$

i.e., the matrix B is more "diagonal" than A. The value of (i,j) is determined according to some ordering, to be determined such that all the off-diagonal elements will be annihilated once in any group of $n(n-1)/2$ rotations (called a "sweep"). A well known example is the cyclic-by-rows ordering, illustrated here in the $n=4$ case:

$$(i,j) = (1,2),(1,3),(1,4),(2,3),(2,4),(3,4). \tag{27}$$

A Jacobi-SVD algorithm for A is simply

Algorithm SVD.

do until convergence
  for each (i,j) according to some preferred ordering
    $A \leftarrow J_{ij}^T A \ K_{ij}$. □

By convergence we mean that the parameter $off(A)$ has fallen below some preselected tolerance. However, it is difficult to monitor $off(A)$ in the settings of parallel computations. Since convergence is fast (ultimately quadratic) it is a usual practice to stop iterations after a sufficiently large number (say ten) of sweeps.

Square array

A "parallel" ordering that allows $\lfloor n/2 \rfloor$ simultaneous rotations was introduced by Brent and Luk[5]. Their new ordering is amply illustrated by the $n = 8$

case:

$$
\begin{aligned}
(p,q) = \; & (1,2)\,,\ (3,4)\,,\ (5,6)\,,\ (7,8)\,, \\
& (1,4)\,,\ (2,6)\,,\ (3,8)\,,\ (5,7)\,, \\
& (1,6)\,,\ (4,8)\,,\ (2,7)\,,\ (3,5)\,, \\
& (1,8)\,,\ (6,7)\,,\ (4,5)\,,\ (2,3)\,, \\
& (1,7)\,,\ (8,5)\,,\ (6,3)\,,\ (4,2)\,, \\
& (1,5)\,,\ (7,3)\,,\ (8,2)\,,\ (6,4)\,, \\
& (1,3)\,,\ (5,2)\,,\ (7,4)\,,\ (8,6)\,.
\end{aligned}
$$

Rotation pairs associated with each "row" of the above ordering can be calculated concurrently. Brent et al.[6] propose a square array of $O(n^2)$ processors implementing a parallel SVD algorithm for an $n \times n$ matrix A:

Algorithm SVD1.

do until convergence
  for each (i,j) according to the "parallel" ordering
    $A \leftarrow J_{ij}^T A\, K_{ij}$.
    { "inner rotations" are used } □

Details on the processor array are given in Brent et al.[5,6]. Important points worth emphasizing are that only nearest neighbor connections are required, that broadcasting can be avoided through a staggering of computations, and that one sweep of the algorithm is implementable in time $O(n)$.

Numerical experiments were performed on a VAX-11/780 at Cornell University. Double floating data types were used: each number is binary normalized, with an 8-bit signed exponent and a 57-bit signed fraction whose most significant bit is not represented. The accuracy is thus approximately 17 decimal digits. The results are presented in Table 1. We started with random $n \times n$ matrices whose elements came from a uniform distribution in the interval (-1,1); we stopped when the parameter $off(A)$ had been reduced to 10^{-12} times its original value. The rate of convergence was quadratic, confirming theoretical predictions, and only eight or fewer sweeps were required for $n \leq 200$. The SVD of an $n \times n$ matrix is thus computable in effectively $O(n)$ time.

Table 1. Average Number of Sweeps
Required by Algorithm SVD1

n	trials	#sweeps
10	1000	4.55
20	100	5.54
30	100	6.09
40	100	6.40
50	100	6.72
80	30	7.30
100	10	7.56
150	3	7.73
200	1	8.10

Triangular array

Luk[9] proposes a triangular processor array that directly computes an SVD of a rectangular matrix. The associated SVD algorithm has two stages. First, a QR-decomposition is computed of A as it is fed into the array. This procedure is quite similar to that of Gentleman-Kung[18]. A major difference is that Luk performs 2×2 QRDs, whereas Gentleman and Kung annihilate individual elements. Second, a Jacobi-SVD algorithm is applied to the resultant triangular matrix. The pivot block is restricted to contiguous diagonal elements, so as to preserve the triangular structure of the matrix. "Outer rotations" are required to ensure that all off-diagonal elements will be annihilated. Details on the array are presented in Luk[9]. Again the important points concern the nearest neighbor connections, the avoidance of broadcast, and the completion of a sweep in $O(n)$ time. We present here the associated SVD algorithm for an $n\times n$ upper triangular matrix A:

Algorithm SVD2.

do until convergence
begin
{ "outer rotations" are required }
for $i = 1, 3, \cdots$ (i odd) do
$A \leftarrow J_{i,i+1}^T\ A\ K_{i,i+1}$;
for $i = 2, 4, \cdots$ (i even) do
$A \leftarrow J_{i,i+1}^T\ A\ K_{i,i+1}$
end. □

Simulation experiments were performed under the same conditions as reported in the previous section. However, we started with upper triangular matrices and

scaled them so that initially $off(A) = 1$. The parameter ϵ represents the machine precision and approximately equals 1.4×10^{-17}. The ultimate quadratic convergence rate of a Jacobi algorithm is nicely exhibited in Table 2.

Table 2. $Off(A)$ After Each Sweep of Algorithm SVD2

n	Sweep				
	1	2	3	4	5
4	1e-01	1e-03	2e-09	$< \epsilon$	
6	3e-02	4e-05	1e-11	$< \epsilon$	
8	1e-01	4e-04	1e-09	$< \epsilon$	
10	1e-01	1e-02	3e-07	$< \epsilon$	
12	2e-02	2e-03	5e-05	1e-11	$< \epsilon$
14	5e-02	2e-03	2e-05	1e-10	$< \epsilon$
16	3e-02	1e-03	6e-06	3e-11	$< \epsilon$
18	1e-01	1e-03	4e-06	2e-10	$< \epsilon$
20	9e-02	5e-03	2e-04	2e-07	1e-13

Missized problems

We conclude with some remarks about the handling of SVD problems whose dimensions differ from the effective dimension of the processor array. To fix the discussion, suppose that A is an $n \times n$ matrix whose SVD we want and that our array can handle SVD problems with maximum dimension N.

If $n < N$, it is natural to have the array compute the SVD of

$$\hat{A} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}, \tag{28}$$

so that

$$\hat{U}^T \hat{A} \hat{V} = \mathrm{diag}(\sigma_1, \cdots, \sigma_n, 0, \cdots, 0) . \tag{29}$$

Brent et al.[6] show how one may take the precaution to ensure

$$\hat{U} = \begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix} \quad \text{and} \quad \hat{V} = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}, \tag{30}$$

whence $U^T A V = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$. Let us point out that an SVD procedure needs not produce matrices $\hat{U}$ and $\hat{V}$ with the above block structure in the case $\mathrm{rank}(\hat{A}) = \mathrm{rank}(A) < n$. For example, if $N = 3$ and $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, then one of the infinitely many SVDs of $\hat{A}$ is

$$\begin{bmatrix} p & p^2 & p^2 \\ p & -p^2 & -p^2 \\ 0 & -p & p \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p & p^2 & p^2 \\ p & -p^2 & -p^2 \\ 0 & -p & p \end{bmatrix} = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{31}$$

where $p = 1/\sqrt{2}$. Further computations are thus necessary before an SVD of A can be obtained.

Next, let us examine how oversized SVD problems may be handled. Partition the matrix A so that

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1k} \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ A_{k1} & \cdots & A_{kk} \end{bmatrix}, \tag{32}$$

where each A_{ij} is $N/2 \times N/2$. (Assume that N, the dimension of the systolic array, is even so that $n = kN/2$.) One way to compute an SVD of A is a "block" Jacobi scheme[6]. In this scheme we repeatedly pick (i,j) satisfying $1 \leq i < j \leq k$ and use an SVD array to solve the $N \times N$ problem:

$$\begin{bmatrix} U_{ii} & U_{ij} \\ U_{ji} & U_{jj} \end{bmatrix}^T \begin{bmatrix} A_{ii} & A_{ij} \\ A_{ji} & A_{jj} \end{bmatrix} \begin{bmatrix} V_{ii} & V_{ij} \\ V_{ji} & V_{jj} \end{bmatrix} = \begin{bmatrix} D_i & 0 \\ 0 & D_j \end{bmatrix}. \tag{33}$$

We then construct an $n \times n$ orthogonal matrix U so that it is equal to the identity matrix except for the four strategic blocks in the (i,i), (i,j), (j,i) and (j,j) positions. Those blocks assume the values as given by (33). An $n \times n$ orthogonal matrix V is constructed in an identical manner. Then the matrix $B = U^T A V$ will have the property that

$$\begin{aligned} off(B) = off(A) &- \|A_{ij}\|_F^2 - \|A_{ji}\|_F^2 \\ &- off(A_{ii}) - off(A_{jj}) . \end{aligned} \tag{34}$$

The indices (i,j) may be chosen according to either Algorithm SVD1 or SVD2. We can exploit a block systolic array, where the diagonal arrays perform SVDs and the off-diagonal arrays matrix-matrix multiplications. The blocks A_{ij} will move around an array of arrays in exactly the same fashion as the elements a_{ij} do in an array of processors. This "block" Jacobi technique will be studied in a future report.

Acknowledgements

The work of the second author was supported in part by the National Science Foundation under grant MCS-8213718.

References

1. G.H. Golub and F.T. Luk, "Singular value decomposition: applications and computations," *Trans. 22nd Conf. Army Mathematicians, ARO Report 77-1* (1977), pp. 577-605.
2. G.H. Golub and C.F. Van Loan, *Advanced Matrix Computations*, The Johns Hopkins Press, Baltimore, 1984.
3. J.J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, 1979.
4. K. Bromley and J.M. Speiser, "Signal Processing Algorithms, Architectures, and Applications," *Tutorial 31, SPIE 27th Annual Internat. Tech. Symp.*, San Diego, 1983.
5. R.P. Brent and F.T. Luk, "The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays," *SIAM J. Sci. Statist. Comput.*, 5 (1984), to appear.
6. R.P. Brent, F.T. Luk and C.F. Van Loan, "Computation of the singular value decomposition using mesh-connected processors," *J. VLSI Computer Systems*, 1 (1984), to appear.
7. A.M. Finn, F.T. Luk, and C. Pottle, "Systolic array computation of the singular value decomposition," *Proc. SPIE Symp. 1982, Vol. 341, Real Time Signal Processing V* (1982), pp. 35-43.
8. D.E. Heller and I.C.F. Ipsen, "Systolic networks for orthogonal equivalence transformations and their applications," *Proc. 1982 Conf. on Advanced Research in VSLI*, MIT, Cambridge (1982), pp. 113-122.
9. F.T. Luk, "A triangular processor array for computing the singular value decomposition," Tech. Report 84-625, Computer Science Dept., Cornell Univ., 1984.
10. R. Schreiber, "A systolic architecture for singular value decomposition," *Proc. 1st Internat. Coll. on Vector and Parallel Comput. in Sci. Applic.*, Paris (1983), pp. 143-148.
11. F.T. Luk, "A Jacobi-like algorithm for computing the QR-decomposition," Tech. Report 84-612, Computer Science Dept., Cornell Univ., 1984.
12. G.W. Stewart, "A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix," Tech. Report 1321, Computer Science Dept., Univ. of Maryland, 1983.
13. F.T. Luk, "Computing the singular-value decomposition on the ILLIAC IV," *ACM Trans. Math. Softw.*, 6 (1980), pp. 524-539.
14. A.H. Sameh, "On Jacobi and Jacobi-like algorithms for a parallel computer," *Math. Comput.*, 25 (1971), pp. 579-590.
15. T.F. Chan, "An improved algorithm for computing the singular value decomposition," *ACM Trans. Math. Softw.*, 8 (1982), pp. 72-83.

16. H.M. Ahmed, J.-M. Delosme and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," *Computer*, 15 no. 1 (Jan. 1982), pp. 65-82.

17. A. Bojanczyk, R.P. Brent and H.T. Kung, "Numerically stable solution of dense systems of linear equations using mesh-connected processors," *SIAM J. Sci. Statist. Comput.*, 5 (1984), pp. 95-104.

18. W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays," *Proc. SPIE Symp. 1981, Vol. 298, Real Time Signal Processing IV* (1981), pp. 19-26.

19. D.E. Heller and I.C.F. Ipsen, "Systolic networks for orthogonal decompositions," *SIAM J. Sci. Statist. Comput.*, 4 (1983), pp. 261-269.

20. L. Johnsson, "A computational array for the QR-method," *Proc. 1982 Conf. on Advanced Research in VSLI,* MIT, Cambridge (1982), pp. 123-129.

21. A.H. Sameh, "Solving the linear least squares problem on a linear array of processors," *Proc. Purdue Workshop on Algorithmically-Specialized Computer Organ.*, W. Lafayette, Sept. 1982.

22. M.R. Hestenes, "Inversion of matrices by biorthogonalization and related results," *J. Soc. Indust. Appl. Math.* 6 (1958), pp. 51-90.